AAGG para la resolución del Pentominó

Dobladez Brisquet, Julia
García Benitez, Arantxa
González Méndez, Álvaro
Millán Palacios, Sandra
Rodríguez Fernández, Cristina
Rojo Gala, Miguel Ángel

Grupo 2

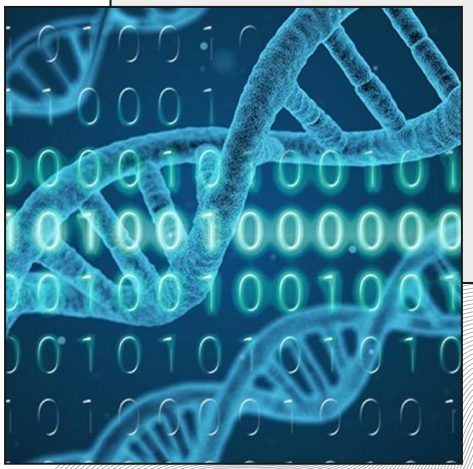
UNIVERSIDAD POLITÉCNICA DE MADRID

POLITÉCNICA

Master Universitario en Inteligencia Artificial

Computación Evolutiva

Índice

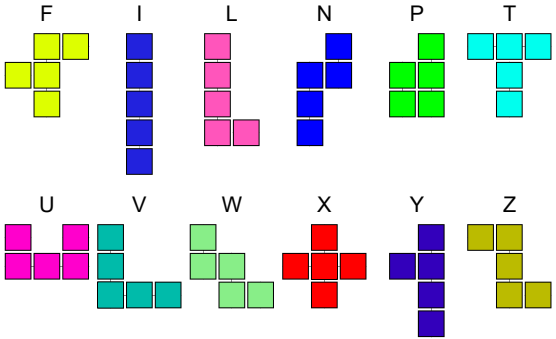


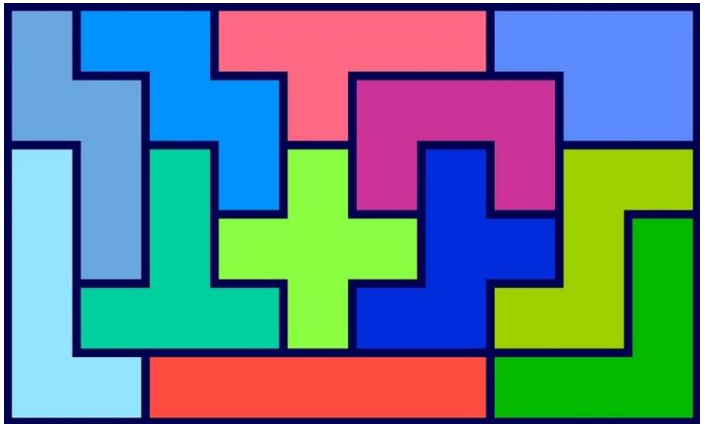
01

Introducción

Pentominó **de Solomon Wolf Golomb**

Un pentominó (pieza) es una figura geométrica compuesta por cinco cuadrados unidos por sus lados.
Existen **12 piezas únicas**, identificadas por letras (**F, I, L, P, N, T, U, V, W, X, Y, Z**).
El objetivo es cubrir un área rectangular de 60 unidades (generalmente un tablero de 6 x 10) utilizando todas las piezas.





Imágenes de Wikipedia

Pentominó

Aunque parece un juego simple, matemáticamente es un problema de Tiling o empaquetado.

Además, hay que cumplir ciertas restricciones :

- Todas las piezas deben usarse exactamente una vez
- No puede haber superposiciones.
- No pueden quedar huecos vacíos.

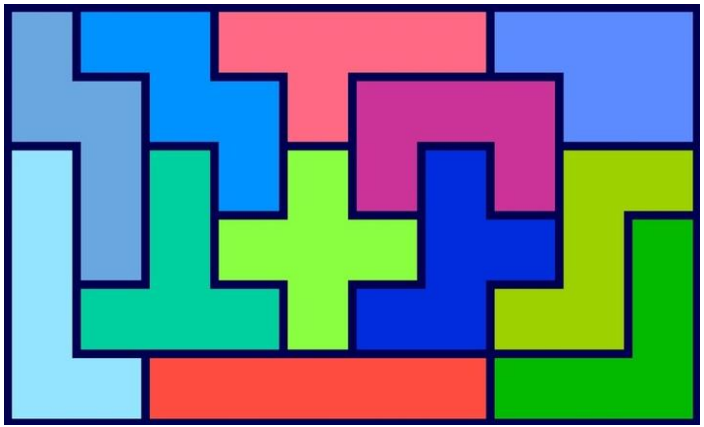
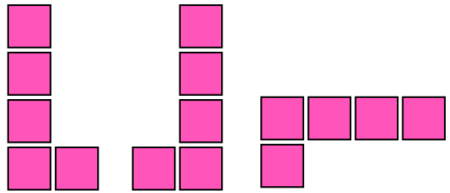


Imagen de Wikipedia

¿Por qué Algoritmos Genéticos?



El Reto Combinatorio: El espacio de búsqueda es inmenso: **12! (permutaciones) x 8^12 (orientaciones posibles)**. → Los métodos deterministas son computacionalmente demasiado costosos sin heurísticas avanzadas.

Ventajas de los AAGG:

- **Exploración Global:** Evitan quedarse atrapados en mínimos locales
- **Optimización indirecta:** En lugar de colocar píxeles, evolucionamos una estrategia de construcción (orden de piezas y rotación).
- **Adaptabilidad:** La función de fitness guía la búsqueda penalizando configuraciones que generan huecos inalcanzables.

Los AAGG permiten transformar un problema de búsqueda exhaustiva en un problema de optimización eficiente.

Para el rectángulo de 6×10 existen exactamente 2339 soluciones

02

Estado del arte

Otros métodos clásicos

Dancing Links (DLX)

Knuth, D. E. (2000)

Recubrimiento Exacto → matriz binaria:

- **Filas** = Cada posible ubicación y rotación (pieza)
- **Columnas** = 60 celdas + las 12 restricciones de "pieza usada". Selecciona filas tal que cada columna = 1

Backtracking: si una pieza no encaja, el algoritmo deshace la colocación para probar la siguiente

Programación Entera

Fabarisova, A. I., & Kartak, V. M. (2021)

- **Modelo determinista** para maximizar la cobertura del área.
- Utiliza variables **binarias** para la selección de piezas (1 = pieza colocada)
- **Restricciones**: No solapamiento (suma de piezas <=1) y suma de usos de cada pieza = 1.

Recocido Simulado

Fabarisova, A. I., & Kartak, V. M. (2021)

- Empieza con un tablero **parcialmente lleno**, y mide la "irregularidad" o **cantidad de huecos**
- Mediante un esquema de enfriamiento, acepta **movimientos que empeoran** el tablero para **escapar** de bloqueos locales y **reordenar** las fichas restantes

Aprendizaje por refuerzo

Fang, J., Rao, Y., Zhao, X., & Du, B. (2023)

- Se formula como un **proceso secuencial** donde el agente decide qué pieza colocar y en qué orientación.
- Cada decisión se evalúa, y el agente recibe **recompensa** si la **colocación es válida** y mejora la cobertura del tablero

CE para juegos similares (Tetris)

Algoritmos Genéticos

Li, Y.-B., Sang, H.-B., Xiong, X., & Li, Y.-R. (2021)

- No evolucionan el tablero directamente, sino los **pesos de una función de evaluación**, la cual mide la calidad del tablero basándose en **alturas y el número de agujeros**.

  Score = w1 (altura) - w2(huecos) - w3(rugosidad)

- El **agente** utiliza esta función en una búsqueda adelantada de un paso para **elegir la colocación** de la pieza que maximice la puntuación del estado subsiguiente.
- Aunque es el **más rápido**, su rendimiento promedio es moderado.

Enfoque Híbrido

Font, J. M., Manrique, D., Larrodera, S., & Ramos Criado, P. (2017)

Las tendencias recientes se basan en **aplicar redes** que **aceleren/evalúen el fitness** de candidatos y usar **poblaciones** con heurísticas humanas.
Un posible enfoque busca combinar la **precisión** de la heurística de búsqueda adelantada de dos pasos con la **velocidad** de las redes neuronales, usando estas últimas como un filtro eficiente.

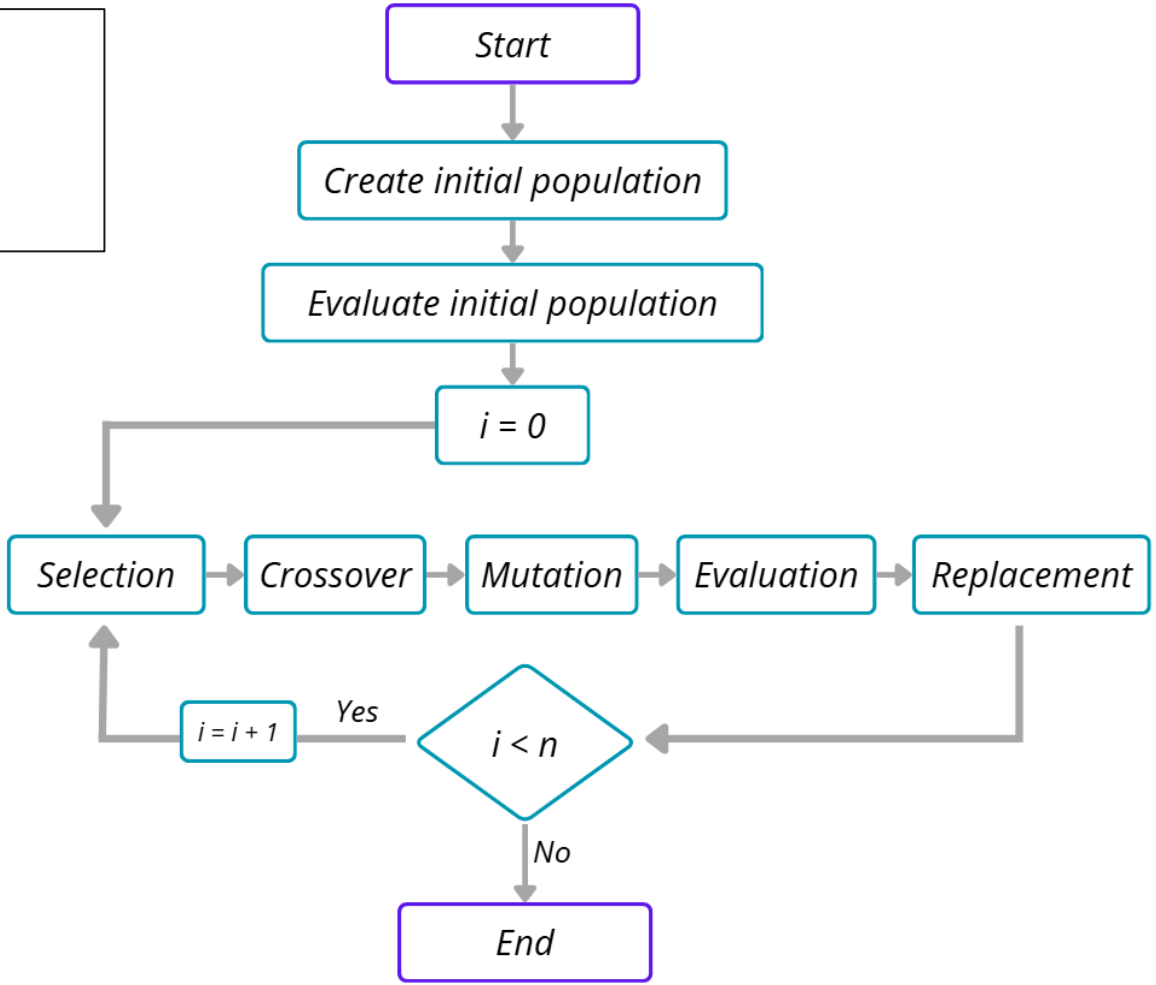
- La **red neuronal** actúa como un filtro o mecanismo de poda: calcula las probabilidades para todas las colocaciones factibles.
- Luego, solo un subconjunto de **colocaciones candidatas** (las que tienen mayor probabilidad) son seleccionadas.

Este enfoque logra el rendimiento promedio más alto con un tiempo de colocación aceptable.

03

Soluciones

Algoritmo Genético



Solución 1

Codificación Binaria

108 bits por individuo --> 12 piezas * 9 bits por pieza

Gen	Conjunto Posibles Valores	Significado	Bits
Reflejar	{0, 1}	{No, Sí}	1
Rotación	{00, 01, 10, 11}	{0°, 90°, 180°, 270°}	2
Posición	{0, … , 63}	Si el valor se encuentra entre el 60 y el 63 no se posiciona	6

Índices de posición en el tablero

50	51	52	53	54	55	56	57	58	59
40	41	42	43	44	45	46	47	48	49
30	31	32	33	34	35	36	37	38	39
20	21	22	23	24	25	26	27	28	29
10	11	12	13	14	15	16	17	18	19
00	01	02	03	04	05	06	07	08	09

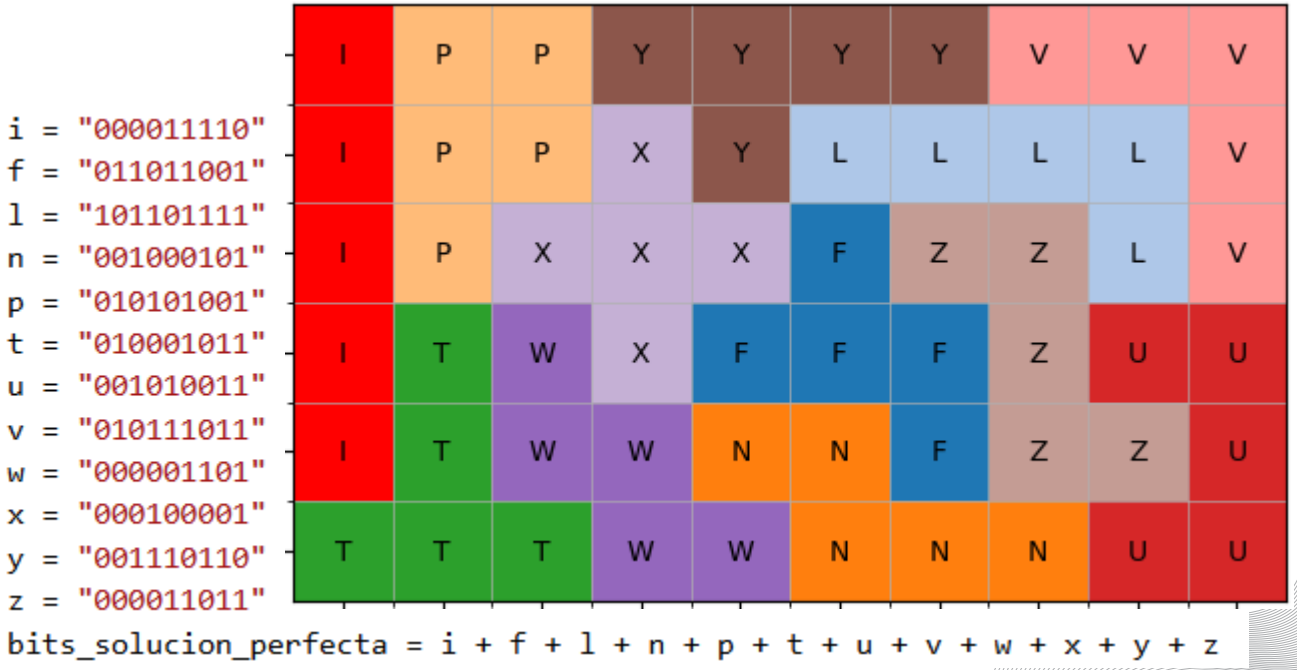
Ejemplo para la pieza L:

0 11 010000

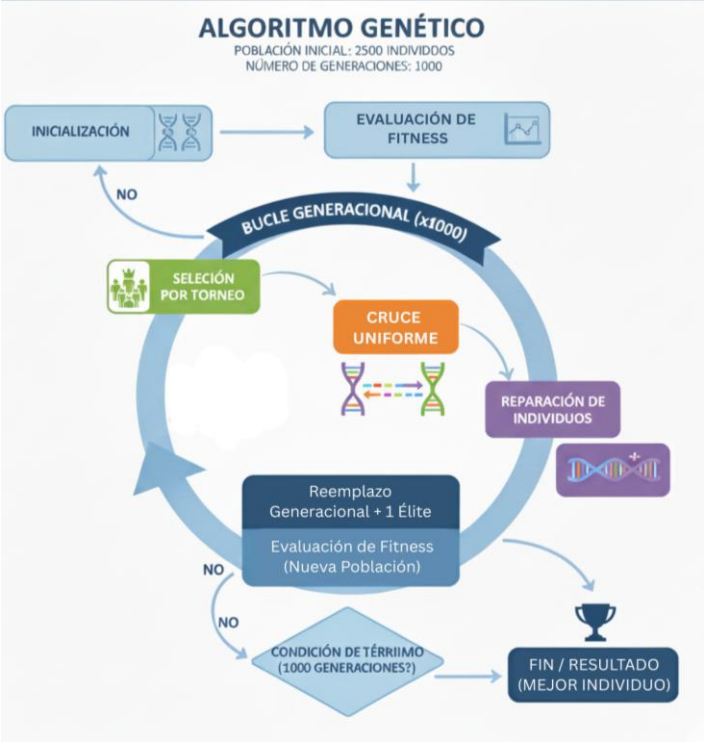
No se refleja, se rota 270º y se construye desde la posición 16

Solución Óptima

```
000011110 011011001 101101111 001000101 010101001 010001011 001010011
          010111011 000001101 000100001 001110110 000011011
```



Ciclo Evolutivo



Generado por Gemini.

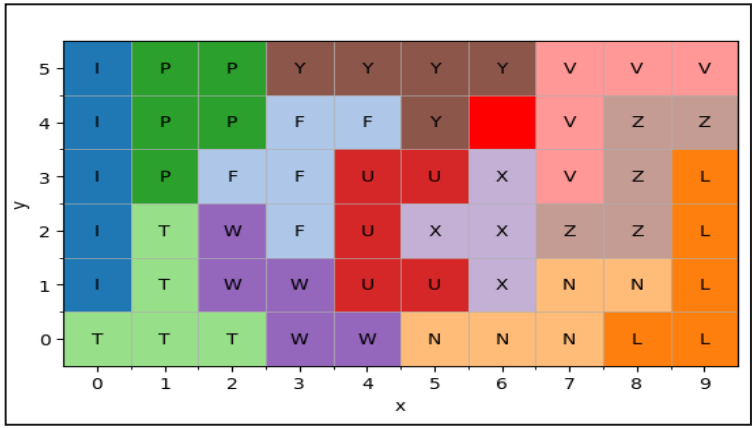
Función Objetivo:

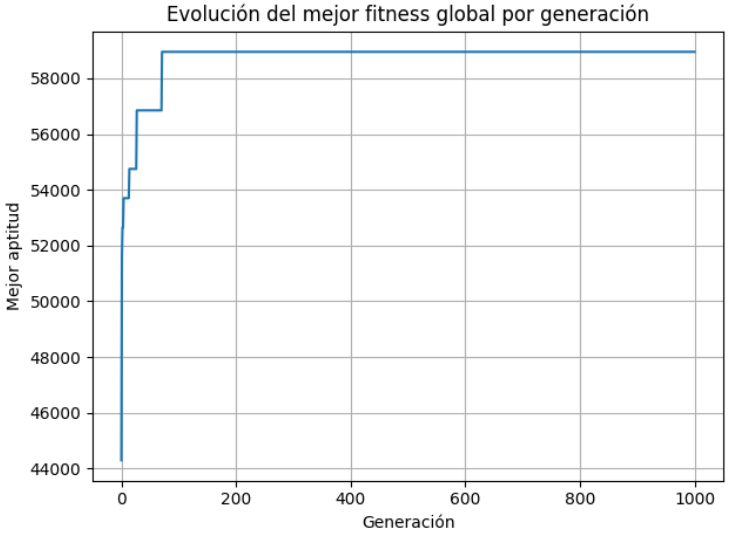
$$\text{Max Fitness} = 1000 \cdot V - 50 \cdot S - 50 \cdot F - 200 \cdot N$$

- V: número de celdas válidas cubiertas por piezas
- S: número de solapamientos
- F: número de piezas fuera del tablero
- N: número de piezas no colocadas

Probabilidad Cruce: 0.9

Resultados





- Tiempo de ejecución: 20 minutos
- El valor de fitness es 58950
- Se han colocado 12 piezas
- No se encuentra la solución óptima, y la solución encontrada no es válida.

¿Mejoras?

✅ Ventajas

- Representación compacta y uniforme
- Fácil de manipular por el GA
- Explora configuraciones muy diversas
- Cálculo rápido

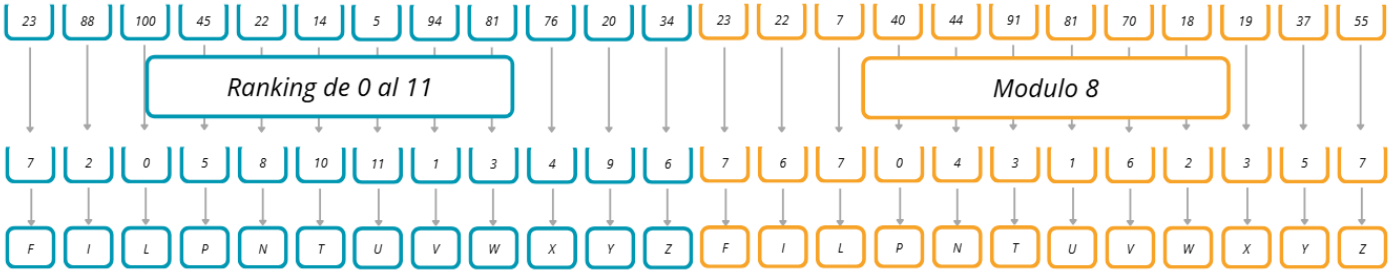
❌ Desventajas

- Espacio de búsqueda enorme
- Alta tasa de soluciones inválidas
- Genes no equivalentes
- Fuerte penalización implica un paisaje de fitness abrupto
- No hay información estructural de la geometría
- Piezas interactúan fuertemente entre sí
- Cruce uniforme puede destruir configuraciones casi válidas

Solución 2

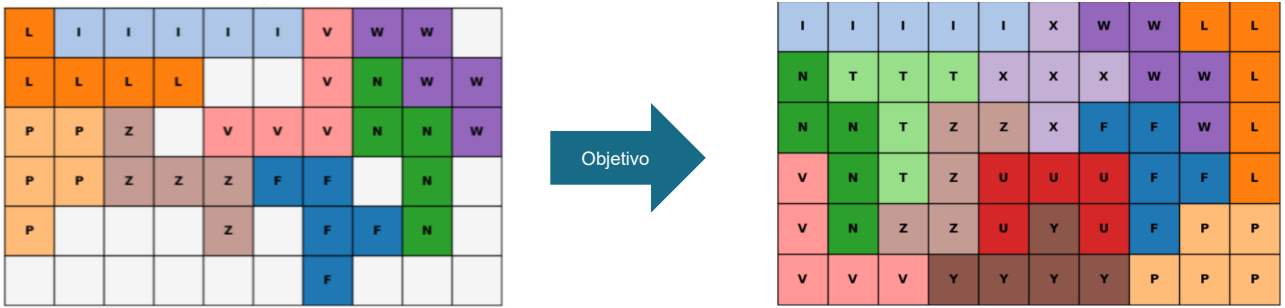
Codificación $(p_F, p_I, p_L, p_P, p_N, p_T, p_U, p_V, p_W, p_X, p_Y, p_Z, r_F, r_I, r_L, r_P, r_N, r_T, r_U, r_V, r_W, r_X, r_Y, r_Z)$

Genotipo: 24 Genes (para 12 piezas) - prioridad + rotaciones



*Ej pieza: **F** - prioridad 7 - rotación 7*

Fenotipo



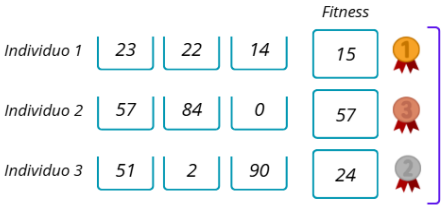
Ciclo evolutivo

Función Objetivo

Fitness (minimizar) = 100 - (piezas_colocadas *5) y si huecos = 0 Fitness = -1000

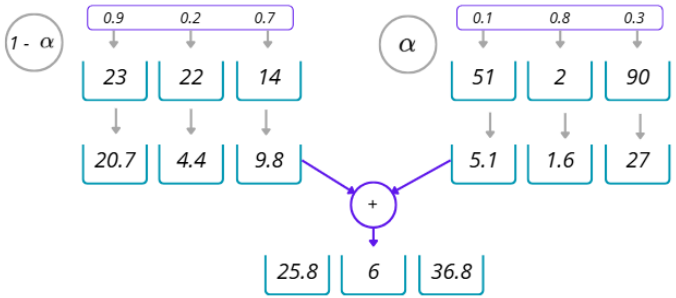
Población 2500 individuos + reemplazo generacional+ Hall of Fame (mejor individuos) + 1000 generaciones

Selección por torneo



El mejor es el Individuo 1

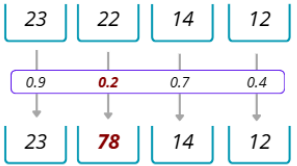
Cruce Blend



Intervalo definido por los padres

Intervalo de cruce

Mutación Uniforme



Umbral

indpb = 0.3

Probabilidad de mutación:

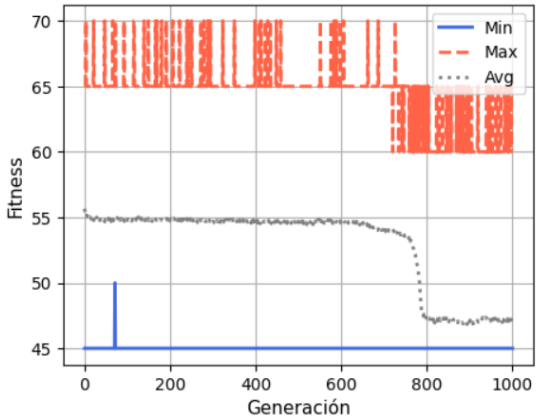
- **Individuo: 0,15**
- **Gen: 0,15**

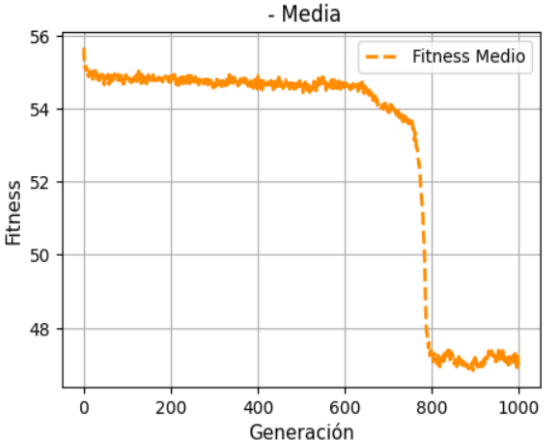
Probabilidad de cruce: 0,9

Resultados

- Se han colocado las 11 piezas
- El valor de fitness es 45
- No logra sacar ninguna configuración óptima

- Convergencia inestable a un óptimo local (11 fichas)
- Gran mejora media en las últimas iteraciones





¿Mejoras?

✅ Ventajas

- Codificación simple y fácil de implementar
- Rápido de evaluar y decodificar.
- Cruce BLX-α aporta alta diversidad inicial
- No solapamientos de piezas + ni piezas fuera del tablero

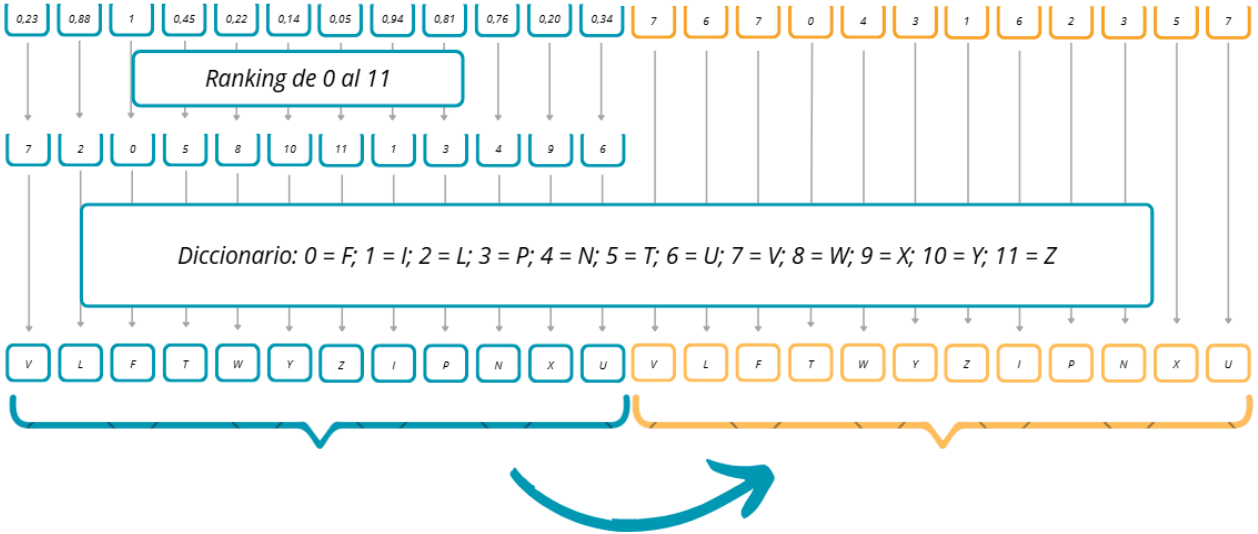
❌ Desventajas

- No controla directamente la posición de las piezas.
- Alta probabilidad de quedar bloqueado en la última pieza.
- **Cambios pequeños en el genotipo producen cambios caóticos en el fenotipo.**
- Función objetivo demasiado simple.
- Fitness no penaliza adecuadamente huecos problemáticos.
- Población converge en soluciones incompletas (11 piezas).
- Diversidad insuficiente tras muchas generaciones.

Solución 3

Codificación (f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, r1, r2, r3, r4, r5, r6, r7, r8, r9, r10, r11, r12)

Genotipo: 24 Genes (para 12 piezas) - figuras + rotaciones



Fenotipo:

Al decodificar se barre de izquierda a derecha y de arriba a abajo seleccionando la ficha en el orden de las f. Si no puede colocar la ficha, pasa a la siguiente y asi hasta llegar al final de la cola (f). Si no se puede deja hueco en blanco y pasa al siguiente (tambien vuelve a la cabeza de la cola de las figuras).

Modelado

Función Objetivo

$$\text{Fitness (minimizar)} = (H \cdot 100) + (I \cdot 50) - (P \cdot 10)$$

- H (Huecos): Penalización por cada casilla vacía.
- I (Islas): Penalización extra calculada mediante Flood Fill. Un grupo de huecos conectados cuenta como 1 isla, por lo tanto varios huecos dispersos cuentan como más islas (peor puntuación).
- P (Piezas): recompensa por cada pieza colocada (para desempatar en caso de empate).

Operadores de variación

Cruce Vinculado

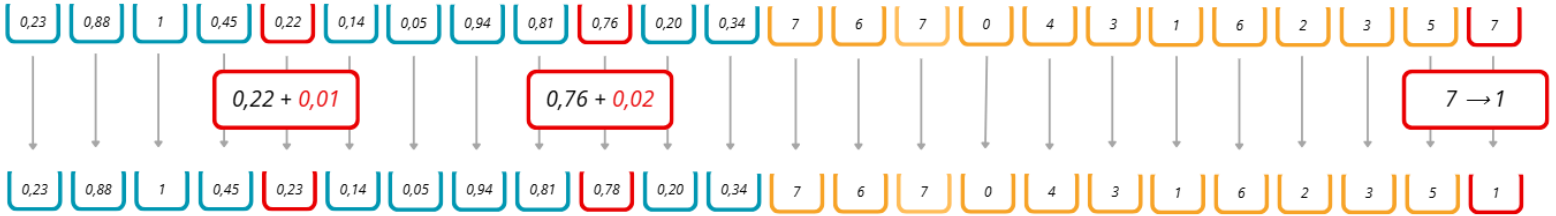
- Problema del cruce estándar: Al cortar un cromosoma por la mitad, se puede heredar la prioridad de un padre y la rotación del otro para la misma pieza → combinaciones que funcionan bien se rompen

Operadores de variación

Mutación Específica

Mutación específica para cada parte:

- Para las Prioridades (Genes 0-11): Se suma un valor aleatorio (pequeño) al gen → Ajustar finamente el orden relativo de las piezas.
- Para las Rotaciones (Genes 12-23): Se cambia el valor por un nuevo entero aleatorio → Probar radicalmente una nueva orientación para la pieza.



AAGG

Hiperparámetros

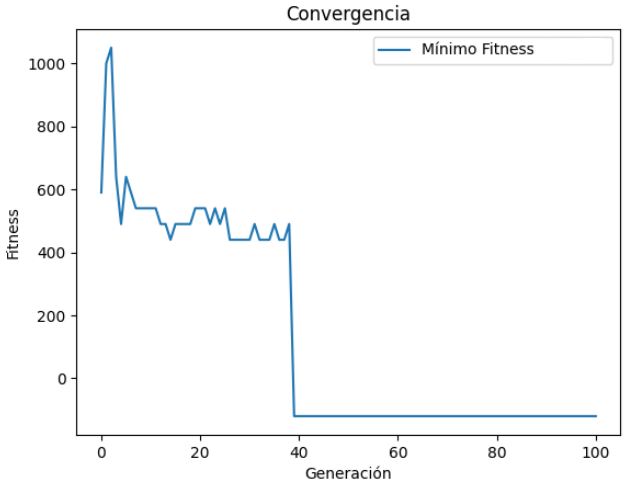
- Población: 500 Individuos (Alta diversidad inicial).
- Generaciones: 100 (Convergencia rápida).
- Probabilidad de Cruce: 0.8 (alta).
- Probabilidad de Mutación gen : 0.05 (muy baja)
- Probabilidad de Mutación individuo : 0.1 (baja)
- Selección: Torneo (Tamaño 3)

Resultados

N	N	Y	Y	Y	Y	X	V	V	V
T	N	N	N	Y	X	X	X	W	V
T	T	T	F	Z	Z	X	W	W	V
T	F	F	F	Z	L	W	W	U	U
P	P	F	Z	Z	L	L	L	L	U
P	P	P	I	I	I	I	I	U	U

- Se han colocado las 12 piezas y no hay huecos vacíos.
- El valor de fitness es -120.0
- Esta es una solución óptima, ya que logra empaquetar todas las piezas sin dejar espacios vacíos.

- Se muestra la evolución del mínimo fitness a lo largo de las generaciones
- El fitness empieza alto (malo) y fluctúa.
- Después de 40 generaciones, el algoritmo es capaz de encontrar la solución óptima.



04

Resultados y Conclusiones

Resultados globales

	Solución óptima	Propiedad del Cierre	Nº generaciones	Tamaño de la población	Hiperparámetros
Solución 1	No	No	1000	2500	• Selección por torneo: 3 • Prob cruce = 0.9 • Prob mutación = 0
Solución 2	No	Si	1000	2500	• Selección por torneo: 3 • Prob cruce = 0.9 • Prob mutación gen = 0.15 • Prob mutación ind = 0.15
Solución 3	Si	Si	100	500	• Selección por torneo: 3 • Prob cruce = 0.8 • Prob mutación gen = 0.05 • Prob mutación ind = 0.1

Conclusiones

- Se ha realizado un estudio **comparativo de distintas codificaciones** (codificación binaria y codificación real).
- Se ha realizado una **aplicación práctica** de los distintos **operadores de variación (mutación y cruce)** estudiados en la asignatura y evaluación de su impacto en la diversidad y convergencia del algoritmo.
- Aunque ciertas configuraciones **lograron alcanzar la solución óptima global**, hemos visto como otras combinaciones no convergieron al óptimo, ilustrando la importancia de la elección de **operadores** y **parámetros**.
- Además, se investigó el estado del arte del **pentominó** y de juegos afines, como el Tetris, para comprender sus fundamentos, dinámicas y posibles **aplicaciones de AAGG** para resolverlos.

Bibliografía

1) Fabarisova, A. I., & Kartak, V. M. (2021). *Solving irregular polyomino tiling problem using simulated annealing and integer programming*. **Communications in Computer and Information Science, 1476**, 175–183. Springer. https://doi.org/10.1007/978-3-030-86433-0_12

2) Knuth, D. E. (2000). *Dancing links*. arXiv. https://arxiv.org/pdf/cs/0011047

3) Fang, J., Rao, Y., Zhao, X., & Du, B. (2023). *A hybrid reinforcement learning algorithm for 2D irregular packing problems*. **Mathematics, 11**(2), 327. https://doi.org/10.3390/math11020327

4) Font, J. M., Manrique, D., Larrodera, S., & Ramos Criado, P. (2017). *Towards a hybrid neural and evolutionary heuristic approach for playing tile-matching puzzle games*. In **2017 IEEE Conference on Computational Intelligence and Games (CIG)** (pp. 76–79). IEEE. https://doi.org/10.1109/CIG.2017.8080418

5) Li, Y.-B., Sang, H.-B., Xiong, X., & Li, Y.-R. (2021). *An improved adaptive genetic algorithm for two-dimensional rectangular packing problem*. **Applied Sciences, 11**(1), 413. https://doi.org/10.3390/app11010413

6) J. M. Font, D. Manrique, J. Ríos. (2009). Redes de Neuronas Artificiales y Computación Evolutiva. Fundación General de la UPM, Madrid, España.

7) Fletcher, J. G. (1965). A program to solve the pentomino problem by the recursive use of macros. Communications of the ACM, 10.

8) Eagle, A., Kato, T., & Minato, Y. (2019). Solving tiling puzzles with quantum annealing.

Gracias

¿Alguna pregunta?

https://github.com/cristirofer/AAGG_Pentomino